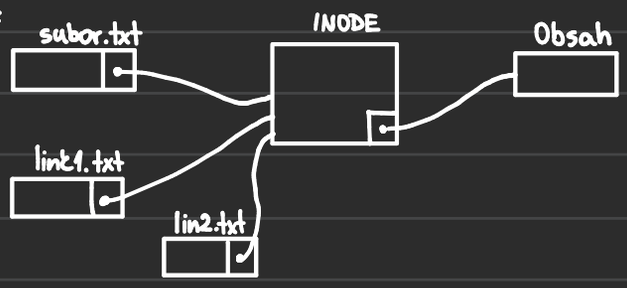
HOMOLA

$15+10+9+12+5=51$

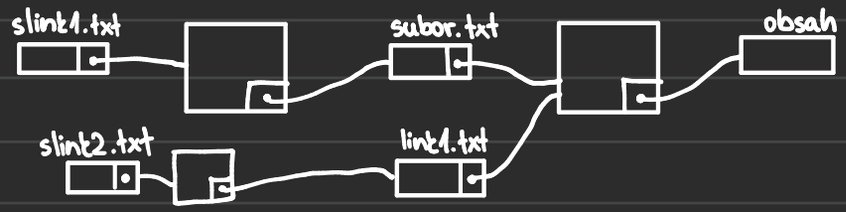
TOMÁŠ HOMOLA

(1) PEVNÝ LINK :



- pri vytvorení pevného linku vlastne priradíme súboru iba nejaké ďalšie meno, čiže ak v tomto prípade zavoláme "cat" + jedno z tých 3 mien, dostaneme ten istý výsledok
- avšak pri vymazaní jedného alebo dvoch z tých mien nám stále ešte ostanú 2 alebo 1 meno, ktoré je asociované so súborom
- jednoducho by som to opísal tak, že vlastne iba zmažeme 1 z 3 referencií na súbor, pričom obsah má potom stále ešte 2 ďalšie

- SYMBOLICKÝ LINK : je vlastne ako smerník na nejaký súbor/adresár, podobne ako smerník v C
- z toho ako tomu rozumiem, by som to znázornil takto :



- zatiaľ čo "link.txt" ukazuje "priamo" na obsah, tak "slink.txt" je vlastne smerník na "subor.txt" a až ten potom je spriahnutý s obsahom a teda ak "subor.txt" zmažeme, tak "slink.txt" stratí cestu k obsahu súboru

TOMÁŠ HOMOLA

- kód:

#1: vytvorí sa súbor s obsahom "hello"

#2 - #5: vytvoria sa pevné linky a cat vypíše do stdout "hello"
↳ prvý obrázok

#6, #7: vytvoria sa symbolické linky

#8: zmaže sa referencia "subor.txt" na obsah

#9: "cat link1.txt" vypíše "hello" → pevný link

#10: tu už by kód nemal fungovať, lebo "slink1.txt" stratil prepojenie "subor.txt" ✓

#11: tu to ešte bude fungovať, vypíše sa "hello"

#12, #13, #14: zmaže sa "link1.txt", "link2.txt" stále funguje (pevný link) ale "slink2.txt" už nebude fungovať ✓

#15 - #19: "mkdir dir" → vytvorenie nového adresára "dir", pevný link
"ldir" by sa nemal dať vytvoriť, lebo adresár môže mať iba symbolický link "sldir" a teda fungovať bude iba, ked napíšeme "ls lsdir" → nič sa ale nevypíše, lebo dir je prázdny

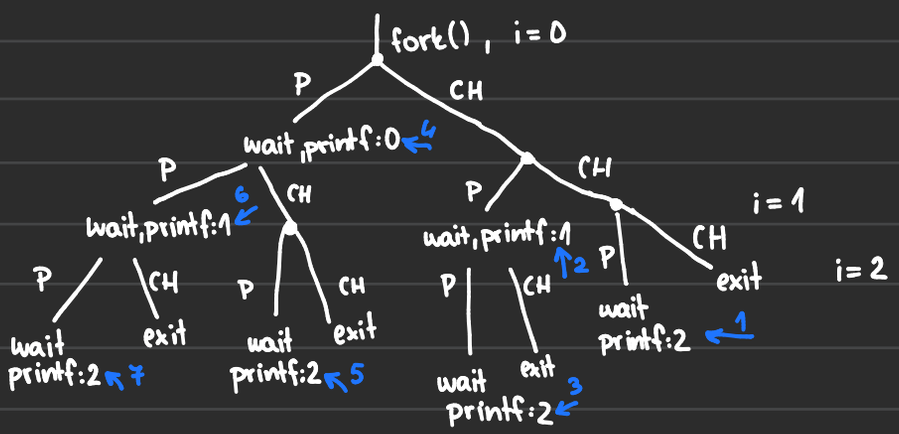
15

(2) # ...

```
main ()
{
   int i, status;

   for (i=0; i<3, i++)
   {
      if (fork() > 0)
         {
            wait (&status)
            print ("%d\n", i);
         }
   }
}
```

- skúsime to nakresliť podobne, ako na prednáške



- prvé teda vypíše 2 (1 →), potom pokračuje na výstupe 1 (3 →), ...
- výstup teda bude zrejme vyzerať takto:

  2, 1, 2, 0, 2, 1, 2 (každé samozrejme v novom riadku)
- modré šípky ukazujú poradie, v ktorom sa vypisujú printf podľa toho, kedy vráti child proces návratovú hodnotu

TOMÁŠ HOMOLA

(3) • regex:

(a) .* baba .* 1 — zbytočné

(b) ^a* 2

(c) *a$ 2

(d) ^[ab]* 2

(e) ^[^a]* 2

(f) ^ nejaký_znak .* ten_istý_znak $

(g) ^ nejaké_slovo .* to_isté_slovo $

nie Aok, správne referencie použiť

9

TOMÁŠ HOMOLA

(4) results.sh:

```
NAME=0
MARK=0
MAIL=0
MSG=0
EMSG=0

while read LINE
do
    NAME=$(echo "$LINE" | cut -f 1 -d ;)
    MARK=$(echo "$LINE" | cut -f 2 -d ;)
    MAIL=$(echo "$LINE" | cut -f 3 -d ;)
    MSG=$(echo "$LINE" | cut -f 4 -d ;)

    EMSG=$(printf "Meno študenta: $NAME \n Známka: $MARK \n Komentár: $MSG ")

    mail "$MAIL" "$EMSG"

done
```

načo?

✓

Ako má ísť na stdin program mail

12

TOMÁŠ HOMOLA

(5) • máme morse_l = ["A .-", ...]

• pre jednotlivé funkcie si spravíme 2 slovníky: slovnik1, slovnik2 z dát, ktoré sú v "mors_l" zozname, kde Key budú buď písmená alebo bodky/čiarky

morse.py:

```
slovnik1 = {'A': '.-', 'B': '-...', ... , ' ': ' '}     # globálne premenné
slovnik2 = {'.-': 'A', '-...': 'B', ... , ' ': ' '}

morse_code(slovo):

    slovo = slovo.upper()
    morse = ""

    for item in slovo:
        morse.append(slovnik1[item] + "/")

    morse.pop()  # vyhodenie posledného "/"

    return morse

morse_decode(slovo)

    slovo = slovo.split("/")
    decoded = ""

    for item in slovo:
        decoded.append(slovnik2[item])

    return decoded
```

ale tie nemáte podľa zadania k dispozícii; máte len morse_l

matúte to.

vrátiť môže reťazec, nie zoznam

takisto.

5